IN THE CLAIMS:

The following listing of claims will replace all prior listings of claims in this application:

1. (Currently Amended): A method for processing a group of related divergent graphics samples in a programmable graphics processing unit having a recirculating pipeline implemented as a single instruction multiple data (SIMD) architecture, the method comprising:

configuring each of a plurality of [[a]] programmable computation units by a field of codewords to perform an operation on multiple samples,

incrementing a subroutine depth of a first sample of the related divergent samples to designate that a first call instruction and a first return instruction are to be executed on the first sample;

determining whether the first call instruction includes a call return that contains a second call instruction and modifying the state data of each of the samples to indicate a number of call returns associated with each of the samples;

pushing state data of a second sample of the related divergent samples upon which the first call and the first return instructions are not to be executed onto a global stack to define the second sample as idle;

dispatching a token associated with the ~~group of~~ samples into the pipeline along with all samples in the group of related divergent samples,

executing the first call instruction and the first return instruction[[s]] on the first sample, but not the second sample; and

storing the processed divergent samples for output or display.

2. (Previously Presented): The method of claim 1, further comprising the step of holding the second sample idle and associating a sample depth in a sample depth score board with the first sample and each sample of the group of related samples, wherein the sample depth represents the number of call/return cycles to be executed on each of the samples.

3. (Previously Presented): The method of claim 2, wherein holding the second sample idle comprises encoding the second sample with non-operation information and

pushing the second sample onto the global stack encoded with information that no operations are to be performed on the second sample, removing the second sample from the group of divergent samples on which operations are performed.

4. (Cancelled)

5. (Previously Presented): The method of claim 1, further comprising the step of incrementing the subroutine depth of the first sample to designate that second call instructions are to be executed on the first sample, and documenting the sample depth of the second sample and testing the sample depth of all samples of the group.

6. (Previously Presented): The method of claim 5, further comprising the step of executing the second call instructions on the first sample.

7. (Currently Amended): The method of claim 1, including comparing the state data associated with each sample in the group of related divergent samples to identify one or more of the samples with the greatest subroutine depth, and removing the identified samples from the working set of data samples in the pipeline which do not have the greatest subroutine depth.

8. (Previously Presented): The method of claim 7, further comprising the step of determining whether an instruction in an instruction sequence includes a call-return that contains the first call instructions, and dispatching a new token into the pipeline with the group of samples after each comparison of the state data.

9. (Currently Amended): A method of processing a group of related divergent graphics samples in a programmable graphics processing unit having a recirculating pipeline embodied as a single instruction multiple data (SIMD) architecture, the method comprising:

configuring each of a plurality of [[a]] programmable computation units by a field of codewords to perform an operation on multiple samples of the groups,

identifying a first sample of the group of related samples having a first subroutine depth;

holding idle a second sample having a second subroutine depth, the first subroutine depth being greater than the second subroutine depth;

dispatching a token associated with the samples of the group into the pipeline along with all the group of samples;

executing operations specified in first return instructions on the first sample;

comparing the sample depth of all the samples of the groups of related samples;

executing an operation specified in the token on samples of the groups of related samples having the greatest subroutine depth; and

storing the processed divergent samples for output or display.

10. (Original): The method of claim 9, wherein holding idle a second sample comprises encoding the second sample with non-operation information.

11. (Currently Amended): The method of claim 9, wherein holding the second sample idle comprises encoding the second sample with non-operation information and pushing the second sample onto the global stack encoded with information that no operations are to be performed on the second sample, and removing the second sample from the group of divergent samples on which operations are performed.

12. (Original): The method of claim 9, further comprising the step of decrementing the first subroutine depth, making the first subroutine depth equal to the second subroutine depth.

13. (Original): The method of claim 12, further comprising the step of determining whether the first subroutine depth is greater than zero.

14. (Original): The method of claim 13, further comprising the step of identifying in an instruction sequence a next instruction to be executed if the first subroutine depth is equal to zero.

15. (Original): The method of claim 13, further comprising the step of executing second return instructions on the first sample and the second sample if the first subroutine depth is greater than zero.

16. (Original): The method of claim 15, further comprising the step of decrementing the first subroutine depth and the second subroutine depth.

17. (Currently Amended): A system for processing a group of related divergent graphics samples in a programmable graphics processing unit having a recirculating pipeline implemented as a single instruction multiple data (SIMD) architecture, the system comprising:

configuring each of a plurality of programmable computation units each of the plurality of programmable computation units configured by a field of ~~codeswords~~ codewords to perform an operation on multiple samples of the groups;

a subroutine depth scoreboard configured to store a subroutine depth corresponding to each sample of the groups of related samples ~~of the~~;

a global stack configured to store state data related to each sample of the group of related samples; and

a remap configured to compare to subroutine depth of each of the samples of the group of related samples and to increment and decrement the subroutine depth in the subroutine depth scoreboard and to push state data onto and to pop state data from the global stack based on the decision as to which of the samples of the group of samples have the greatest subroutine depth.

18. (Previously Presented): The system of claim 17, wherein the remap is further configured to determine that first call instructions are to be executed on first samples selected as having the greatest subroutine depth, but not the second samples of the group identified as not having the greatest subroutine depth, to increment the first subroutine depth to designate that the first call instructions are to be executed on the first samples, and to push state data of the second samples onto the global stack.

19. (Previously Presented): The system of claim 18, wherein the remap is further configured to encode the second samples with non-operation data, to generate a PC token for executing the first call instructions, the PC token containing one or more codewords that configure programmable computation units within a recirculating shader

pipeline to execute the first call instructions, and to dispatch the PC token into the recirculating shader pipeline, followed by the first samples and the second samples.

20. (Previously Presented): The system of claim 17, wherein the remap is further configured to determine that first return instructions are to be executed on the first samples, but not the second samples, to encode the second samples with non-operation data, to generate a PC token for executing the first return instructions, the PC token containing one or more codewords that configure programmable computation units within the recirculating shader pipeline to execute the first return instructions, and to dispatch the PC token into the recirculating shader pipeline, followed by the first samples and the second samples.

21. (Previously Presented): The system of claim 20, wherein the remap is further configured to decrement the first subroutine depth and to pop state data of the second samples from the global stack.

22. (Currently Amended): A system for processing a group of related divergent graphics samples in a programmable graphics processing unit, the system comprising:

configuring each of a plurality of a programmable computation units by a field of codewords to perform an operation on multiple samples of the groups;

means for incrementing a first subroutine depth of a first set of samples of the groups of samples to designate that first call instructions are to be executed on the first set of samples based on identifying the first set of samples having a greater subroutine depth than any sample of the second set of samples;

means for maintaining a score board of subroutine depth for each sample of the groups of related samples;

means for comparing the subroutine depth of every sample of the groups of samples prior to executing each call instruction on any of the samples;

means for pushing state data of a second sample upon which the first call instructions are not to be executed onto a global stack;

means for dispatching all the samples of the groups of samples through the pipeline with a token for configuring the pipeline after each comparison of the subroutine depths of each of the samples;

means for executing the first call instructions on the first sample;

means for executing first return instructions on the first sample;

means for decrementing the first subroutine depth; and

means for popping state data of the second sample from the global stack.

23. (Original): The system of claim 22, further comprising means for holding the second sample idle.

24. (Original): The system of claim 23, wherein means for holding comprises encoding the second sample with non-operation information.

25. (Original): The system of claim 22, further comprising means for determining whether the first subroutine depth is greater than zero and means for executing second return instructions on the first sample and the second sample if the first subroutine depth is greater than zero.